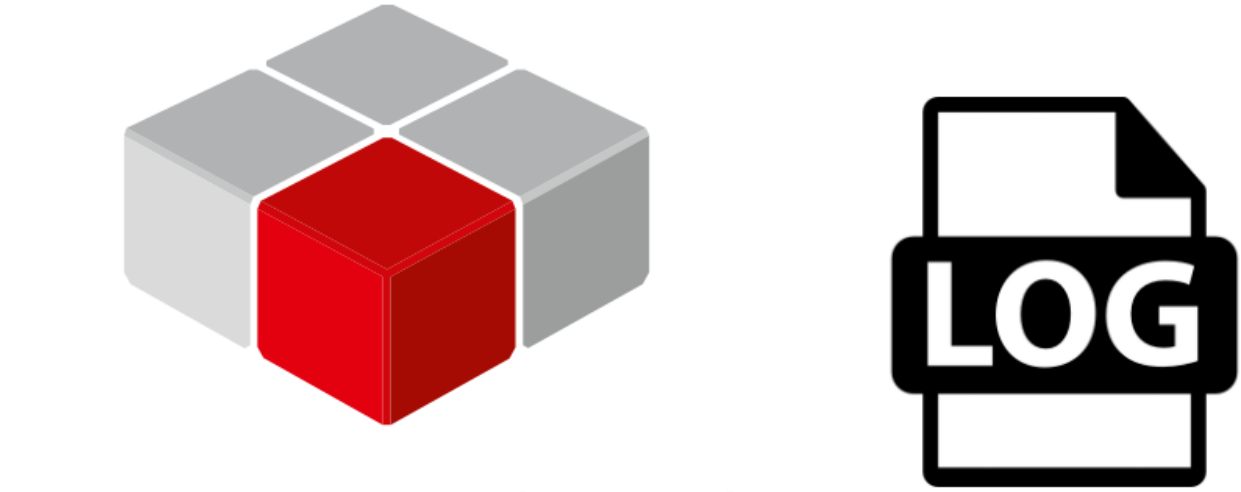
Работа с логами в CODESYS V3



CODESYS

25.03.2024
версия 1.1

https://oscat.ru/

Оглавление

Введение

В процессе отладки и эксплуатации системы автоматизации важную роль играет логирование информации об ошибках и событиях. В ПЛК с системой исполнения CODESYS V3.5 для этих целей используется компонент **PLC Logger** (в русскоязычной локализации – «журнал ПЛК»), основанный на системной библиотеке **CmpLog**. Стандартные компоненты CODESYS используют возможности библиотеки для записи в лог сообщений о своей работе. Кроме того, пользователь может использовать библиотеку для записи в лог собственных сообщений.

С точки зрения внутренней реализации – лог представляет собой набор сообщений, сохраняемых в режиме циклического буфера (то есть самые новые сообщения перезаписывают самые старые). Метод хранения сообщений (бэкенд) зависит от конкретного контроллера – они могут храниться в RAM, во flash-памяти (в виде файлов), отправляться по UDP на syslog-сервер и т.д.

В данном документе рассматривается:

- интерфейс логгера CODESYS (вкладка **Device – Журнал**);
- настройки логгера в конфиг-файле;
- функционал библиотеки CmpLog.

Автор: Евгений Кислов

1. Интерфейс логгера CODESYS

Информация об ошибках и событиях ПЛК отображается в CODESYS на вкладке **Device – Журнал**. Для того чтобы просмотреть журнал – необходимо подключиться к контроллеру. Для каждого события отображаются:

- **Жесткость** – класс события (Предупреждение/Ошибка/Исключение/Сообщение/Сообщение отладки);
- **Временная отметка** – метка времени события. По умолчанию отображается с учетом часового пояса ПЛК, при установленной галочке **UTC-время** – метка времени отображается в UTC+0;
- **Описание** – текст сообщения;
- **Компонент** – название компонента, сформировавшего событие.

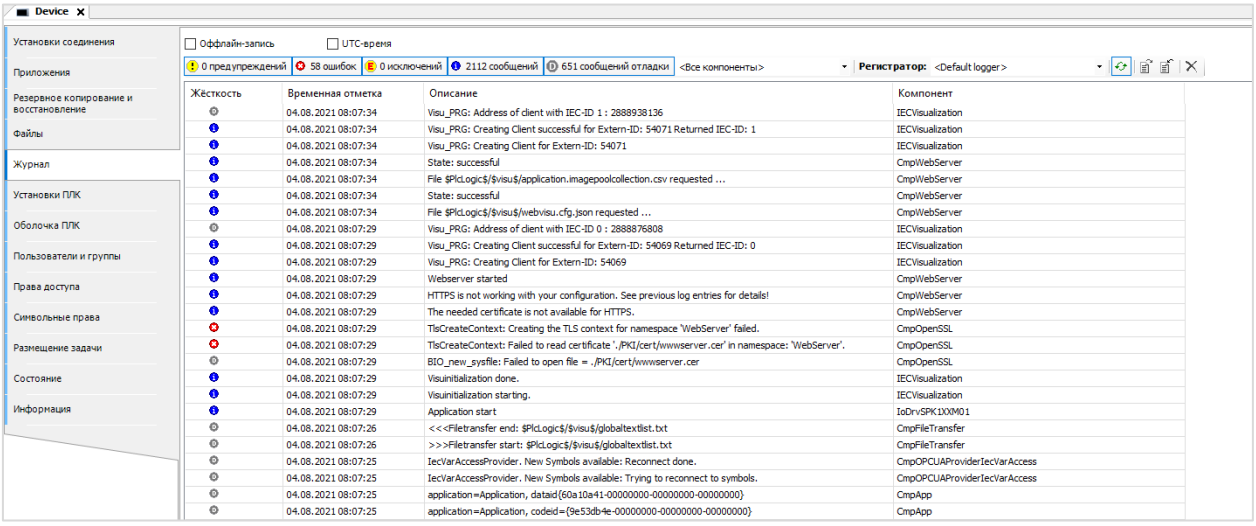


Рис. 1.1. Внешний вид вкладки **Device – Журнал**

С помощью кнопок в верхней левой части вкладки можно настроить фильтрацию событий по классам. С помощью выпадающих списков можно отфильтровать события, относящиеся к конкретному компоненту и логгеру (в контроллере может быть запущено несколько логгеров – например, основной, коммуникационный и т.д.).

Пока актирован режим **Auto scroll** () – при появлении нового события оно будет автоматически отображено в логгере (если режим отключен – то новые события отображаться не будут). Кнопки  используются для экспорта/импорта лога в файл формата .xml. Кнопка очищает окно лога.

В случае установки галочки **Оффлайн-запись** – при отключении от контроллера окно лога будет автоматически очищено.

На вкладке **Device – Журнал** отображается системный журнал ПЛК. В нем фиксируются события, связанные с работой контроллера. Кроме того, отдельные компоненты могут иметь свои журналы. В частности, это касается коммуникационных компонентов – EtherCAT Master, EtherCAT Slave, Modbus TCP Master и т.д.

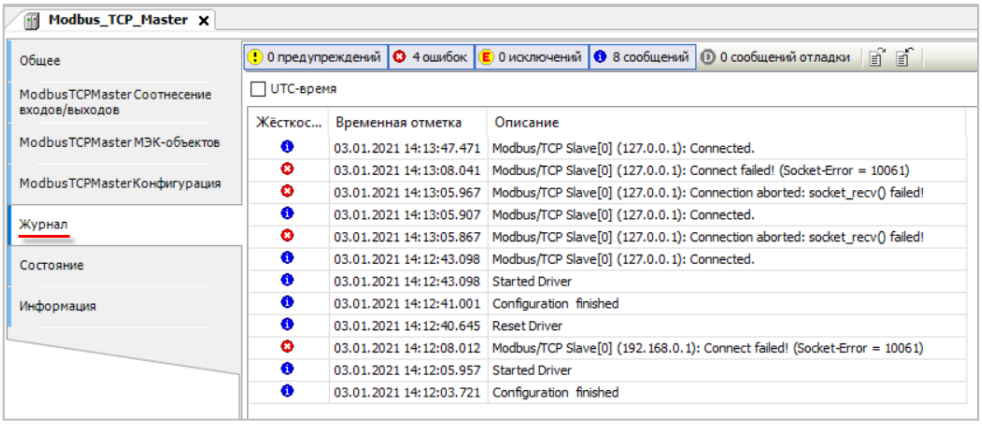
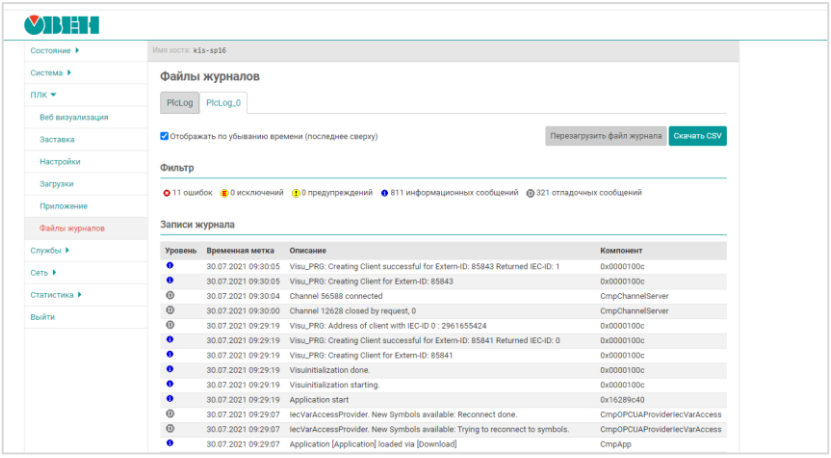


Рис. 1.2. Журнал компонента Modbus TCP Master

Некоторые контроллеры поддерживают отображение лога в web-конфигураторе.



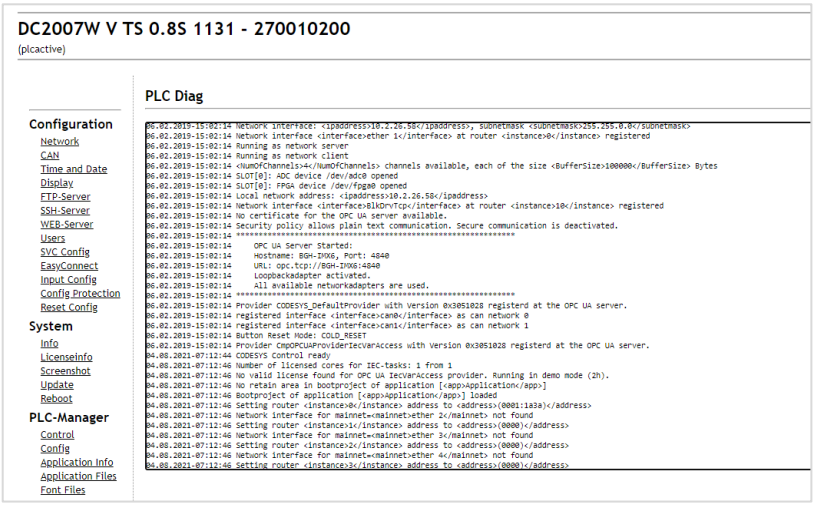


Рис. 1.3. Отображение лога в web-интерфейсе контроллеров ОВЕН и Berghof

2. Настройки логгера в конфиг-файле CODESYS

Настройки системного логгера хранятся в конфиг-файле CODESYS (CODESYSControl.cfg). Они расположены в секции [CmpLog].

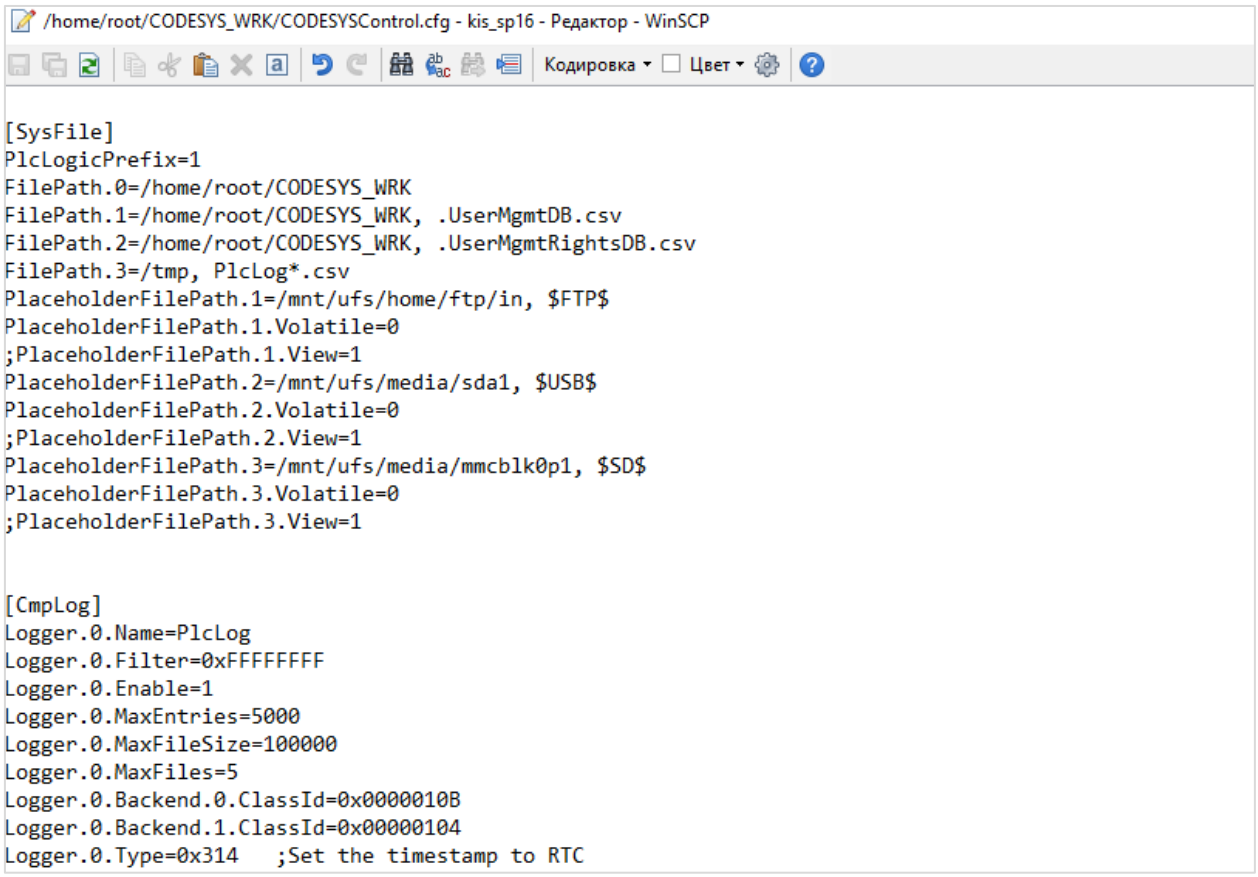


Рис. 2.1. Фрагмент конфиг-файла с настройками логгера

Табл. 2.1. Параметры логгера (секция [CmpLog])

Настройка	Описание
Name	Имя логгера (если логи сохраняются в файлах – то названия файлов состоят из имени логгера и порядкового номера – например, PlcLog_0.csv и т.д.)
Filter	Фильтр классов сообщений, которые поддерживаются логгером. Значение фильтра представляет собой комбинацию флагов LogClass. По умолчанию используется значение **0xFFFFFFFF** (**LOG_ALL**) – в этим режимы логгер поддерживает все классы событий
Enable	**1** – логгер включен, **0** – логгер отключен
MaxEntries	Максимальное число записей в логгере. По достижению этого значения – самые старые записи начинают перезаписываться
MaxFileSize	Максимальный размер файла лога в байтах (используется в том случае, если бэкенд логгера – файлы). При достижении файлом этого размера – создается новый файл лога (см. **MaxFiles**)
MaxFiles	Максимальное число файлов логов (используется в том случае, если бэкенд логгера – файлы). Если создано максимальное число файлов и размер последнего достиг **MaxFileSize** – то начинается перезапись самого старого файла
Backend.<№>.ClassId	Тип бэкенда. Из внутренней документации известно, что значение **0x104** соответствует файлам, **0x135** – отправке на syslog-сервер, а **0x10B** – выводу лога в последовательный порт (через компонент **SysOut**)
Type	Настройки логгера (комбинация флагов LogTypes)

Типовым вариантом бэкенда логгера являются файлы. На рис. 2.1 в секции [SysFile] присутствует строка

FilePath.3=/tmp, PlcLog.csv*

Это означает, что файлы логгера будут сохраняться в директории /tmp.

Начиная с версии **CODESYS V3.5 SP17** в системе доступен специальный логгер **Audit log**. В него записывается информация о действиях пользователя:

- подключении к ПЛК и отключении от него;
- запуске и остановке приложения;
- загрузке проекта и подключении с онлайн-изменением;
- сбросе приложения.

Для активации этого лога необходимо внести соответствующие настройки в конфиг-файл CODESYS (при этом в состав рантайма контроллера должен входить компонент **CmpAuditLog**, иначе данный функционал не будет работать):

```
[CmpLog]
; настройки Logger.0 опущены (см. рис. 2.1)
Logger.1.Name=.Audit.log
Logger.1.Enable=1
Logger.1.Backend.0.ClassId=0x10B   ;writes logger messages in a file
Logger.1.Backend.1.ClassId=0x135   ;sends logger messages as UDP syslog
```

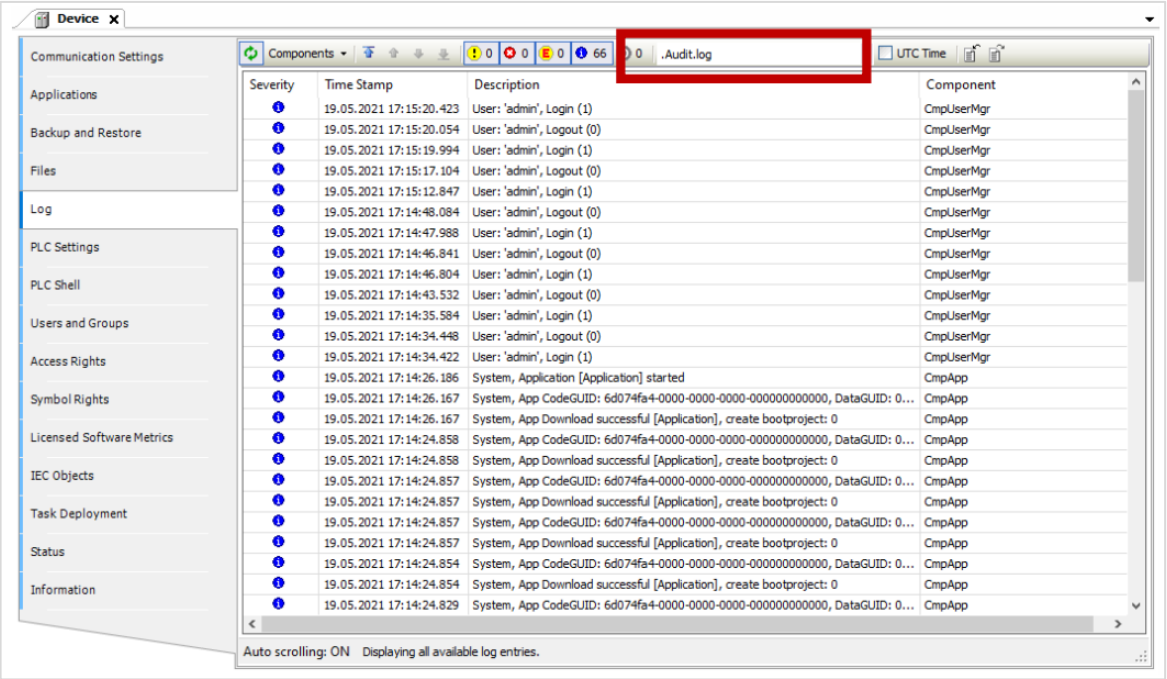


Рис. 2.2. Отображение **Audit log**

3. Библиотека CmpLog

3.0. Основная информация

Библиотека **CmpLog** используется для создания логгеров из кода программы и вывода в них сообщений. Библиотека не позволяет определить бэкенд логгера – поэтому если логгер был создан из кода, то его сообщения будут размещаться в RAM и исчезнут после перезагрузки контроллера.

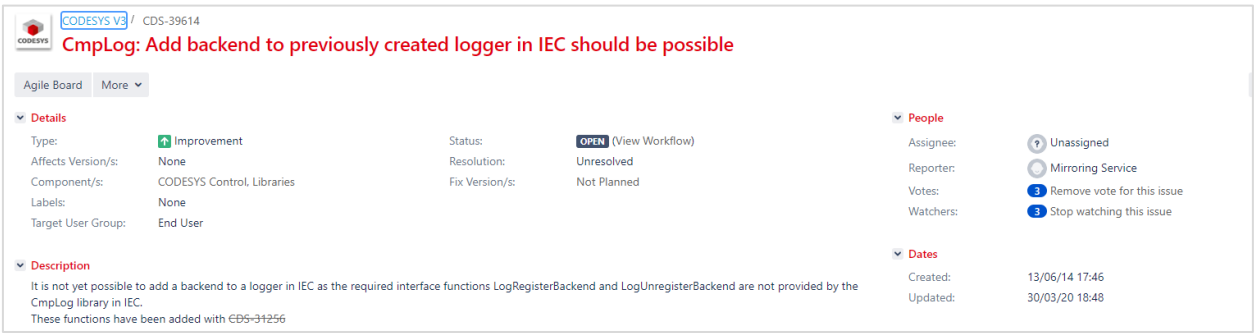


Рис. 3.1. Сообщение в баг-трекере о невозможности добавления бэкенда логгера из кода приложения (CDS-39614)

Библиотека имеет реализацию на Си, которая может использоваться разработчиками контроллеров при создании своих компонентов. В CODESYS доступна одноименная библиотека-обертка, в которой присутствует только часть функций исходной библиотеки.

Общий принцип с работы с логгером похож на работу с файлом и для него используется приблизительно такой же набор функций. Функции библиотеки являются синхронными (т.е. их вызовы являются блокирующими).

3.1. Структура LogOptions

Структура **LogOptions** описывает настройки логгера, передаваемые при вызове функции LogCreate. Список настроек совпадает с настройками в конфиг-файле CODESYS.

Табл. 3.1. Описание переменных структуры **LogOptions**

Переменная	Тип	Описание
szName	STRING(31)	Имя логгера
bEnable	DINT	**1** – логгер включен, **0** – логгер отключен
uiType	UDINT	Настройки логгера (комбинация флагов LogTypes)
uiFilter	UDINT	Фильтр классов сообщений, которые поддерживаются логгером. Значение фильтра представляет собой комбинацию флагов LogClass. По умолчанию используется значение **0xFFFFFFFF** (**LOG_ALL**) – в этим режимы логгер поддерживает все классы событий
iMaxEntries	DINT	Максимальное число записей в логгере. По достижению этого значения – самые старые записи начинают перезаписываться
iMaxFileSize	DINT	Максимальный размер файла лога в байтах (используется в том случае, если бэкенд логгера – файлы). При достижении файлом этого размера – создается новый файл лога (см. **iMaxFiles**). Поскольку структура не позволяет задать бэкенд логгера – то в текущей реализации переменная не имеет смысла
iMaxFiles	DINT	Максимальное число файлов логов (используется в том случае, если бэкенд логгера – файлы). Если создано максимальное число файлов и размер последнего достиг **iMaxFileSize** – то начинается перезапись самого старого файла. Поскольку структура не позволяет задать бэкенд логгера – то в текущей реализации переменная не имеет смысла

3.2. Структура EVTPARAM_CmpLogAdd

Структура **EVTPARAM_CmpLogAdd** содержит информацию о событии, генерируемом при записи нового сообщения в бэкенд логгера. В текущих версиях CODESYS она не может использоваться в коде приложения (только в компонентах, написанных на C):

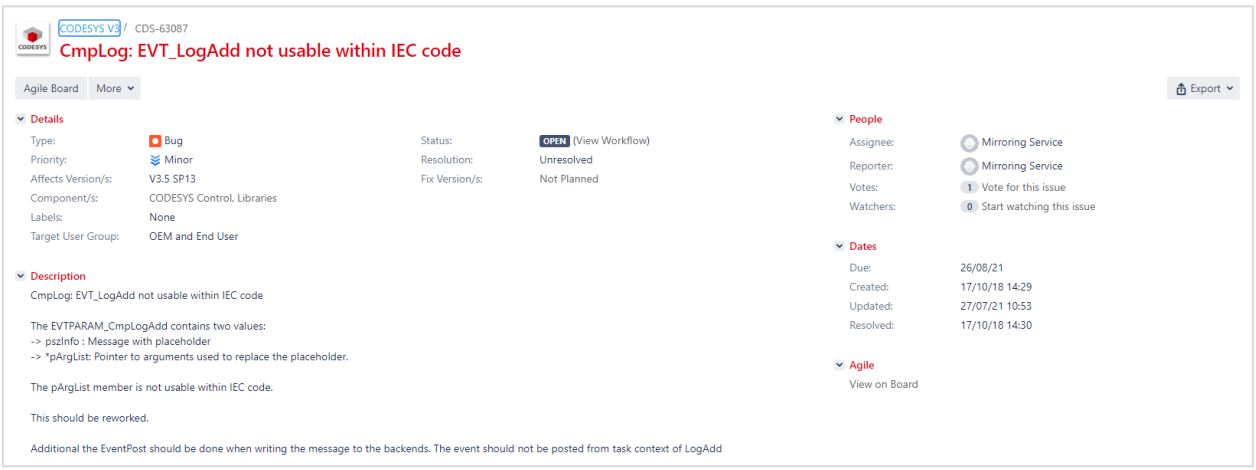


Рис. 3.2. Сообщение в баг-трекере о невозможности использования структуры **EVTPARAM_CmpLogAdd** в коде приложения (CDS-63087)

3.3. Список глобальных констант EventIDs

Список глобальных констант **EventIDs** содержит идентификаторы различных событий, отправляемых компонентом **CmpLog** (не используются в коде приложения).

3.3. Список глобальных констант LogClass

Список глобальных констант **LogClass** описывает классы сообщений логгера.

Табл. 3.1. Описание констант списка **LogClass**

Константа	Тип	Значение	Описание
LOG_NONE	UDINT	16#0	Класс не определен
LOG_INFO	UDINT	16#1	Информационное сообщение
LOG_WARNING	UDINT	16#2	Предупреждение
LOG_ERROR	UDINT	16#4	Ошибка
LOG_EXCEPTION	UDINT	16#8	Исключение
LOG_DEBUG	UDINT	16#10	Сообщение отладки
LOG_USER_NOTIFY	UDINT	16#10000	Особый класс, используемый совместно с одним из других классов (см. примеры ниже). Сообщение такого класса отображается в среде разработки в виде всплывающего окна
LOG_ALL	UDINT	16#FFFFFFFF	Класс, включающий в себя все остальные классы. Может использоваться только при вызове LogCreate (но не может использоваться при вызове LogAdd2, так как не описывает «реальный» класс сообщения)

При вызове функций константы могут комбинироваться с помощью оператора OR. Примеры:

```
// настраиваем поддерживаемые классы для создаваемого логгера.
// uiFilter - поле структуры LogOptions (с экземпляром stLogOptions)
// создаваемый логгер будет поддерживать только сообщения об ошибках и исключениях
stLogOptions.uiFilter := CmpLog.LogClass.LOG_ERROR OR CmpLog.LogClass.LOG_EXCEPTION;


// настраиваем тип сообщения, которое будет записано в лог
// udiClassID будет передан в функцию LogAdd2
// сообщение будет иметь класс «Ошибка», а также отображаться в среде в виде
// всплывающего окна
udiClassID := CmpLog.LogClass.LOG_ERROR OR CmpLog.LogClass.LOG_USER_NOTIFY;
```

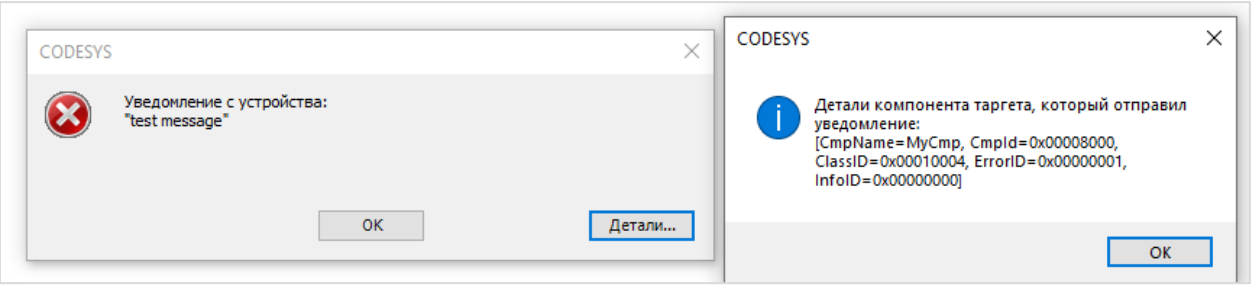


Рис. 3.3. Пример всплывающего окна (LOG_USER_NOTIFY)

3.4. Список глобальных констант LogConstants

Список глобальных констант **LogConstants** описывает псевдонимы для некоторых дескрипторов (хэндлов) логгера. Эти псевдонимы могут передаваться на вход **hLogger** функций библиотеки.

Табл. 3.2. Описание констант списка **LogConstants**

Константа	Тип	Значение	Описание
STD_LOGGER	CmpLog.SysTypes.RTS_IEC_HANDLE	0	Псевдоним стандартного логгера (<default logger> на вкладке **Device – Журнал**)
LOG_STD_LOGGER	CmpLog.SysTypes.RTS_IEC_HANDLE	STD_LOGGER	Аналогично STD_LOGGER
LOG_INVALID_HANDLE	CmpLog.SysTypes.RTS_IEC_HANDLE	RTS_INVALID_HANDLE[1]	Псевдоним некорректного дескриптора (оставлен для обратной совместимости со старыми версиями библиотеки)

3.5. Список глобальных констант LogTypes

Список глобальных констант **LogTypes** описывает настройки логгера. Настройки могут комбинироваться через оператор OR (в тех сочетаниях, в которых это имеет смысл – например, нельзя комбинировать LT_TIMESTAMP_RTC и LT_TIMESTAMP_MS).

Табл. 3.3. Описание констант списка **LogTypes**

Константа	Тип	Значение	Описание
LT_HIGHSPEED	UDINT	16#1	Еще не реализовано.
LT_SAFE	UDINT	16#2	Еще не реализовано (запись лога в RETAIN-память)
LT_NORMAL	UDINT	16#4	Запись лога в RAM
LT_TIMESTAMP_RTC	UDINT	16#10	Метка времени – системное время из RTC с точностью до секунд
LT_TIMESTAMP_RTC_HIGHRES	UDINT	16#2000	Метка времени – системное время из RTC с точностью до миллисекунд
LT_TIMESTAMP_MS	UDINT	16#20	Метка времени – счетчик тиков в миллисекундах
LT_TIMESTAMP_US	UDINT	16#40	Метка времени – счетчик тиков в микросекундах
LT_TIMESTAMP_NS	UDINT	16#80	Метка времени – счетчик тиков в наносекундах
LT_NO_DISABLE	UDINT	16#100	Запрет возможности отключения логгера
LT_DUMP_ASYNC	UDINT	16#200	Запись в бэкенд происходит асинхронно
LT_DUMP_ALWAYS	UDINT	16#400	Запись в бэкенд происходит при появлении каждого нового события
LT_DUMP_ON_CLOSE	UDINT	16#800	Запись в бэкенд происходит при закрытии логгера
LT_DUMP_ON_REQUEST	UDINT	16#1000	Запись в бэкенд происходит при вызове функции **LogDumpEntries** (доступна только в Си-версии библиотеки)
LT_STD	UDINT	см. ниже	Стандартные настройки логгера (см. ниже)

Стандартные настройки логгера (LT_STD) представляют собой следующее сочетание настроек: *(((LT_NORMAL OR LT_TIMESTAMP_RTC) OR LT_NO_DISABLE) OR LT_DUMP_ASYNC)*

[1] С пространством имен – CmpLog.SysTypes.HandleConstants.RTS_INVALID_HANDLE

3.6. Функция LogCreate

Функция **LogCreate** создает новый логгер и возвращает его дескриптор, который может использоваться другими функциями библиотеки. После создания логгер остается открытым до его закрытия функцией LogClose. Записи логгера размещаются в RAM.

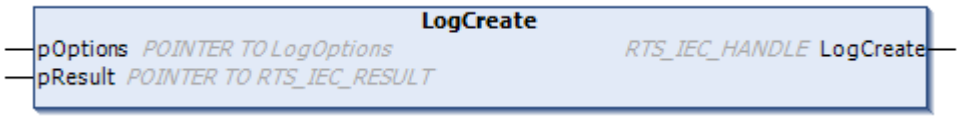


Рис. 3.4. Внешний вид функции **LogCreate** на языке CFC

Табл. 3.4. Описание входов и выходов функции **LogCreate**

Название	Тип	Описание
Входы		
pOptions	POINTER TO LogOptions	Указатель на структуру настроек логгера
pResult	POINTER TO CmpLog.SysTypes.RTS_IEC_RESULT	Указатель на переменную, в которую будет записан код ошибки (см. библиотеку **CmpErrors**)
Выходы		
LogCreate	CmpLog.SysTypes.RTS_IEC_HANDLE	Дескриптор созданного логгера

3.7. Функция LogOpen

Функция **LogOpen** открывает существующий логгер и возвращает его дескриптор, который может использоваться другими функциями библиотеки. После вызова функции логгер остается открытым до его закрытия функцией LogClose.

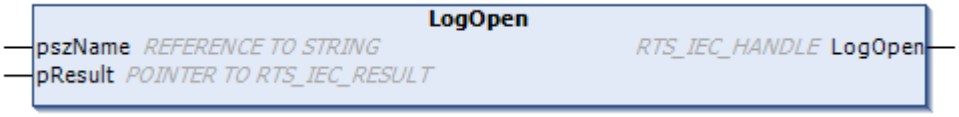


Рис. 3.5. Внешний вид функции **LogOpen** на языке CFC

Табл. 3.5. Описание входов и выходов функции **LogOpen**

Название	Тип	Описание
Входы		
pszName	REFERENCE TO STRING	Имя логгера
pResult	POINTER TO CmpLog.SysTypes.RTS_IEC_RESULT	Указатель на переменную, в которую будет записан код ошибки (см. библиотеку **CmpErrors**)
Выходы		
LogOpen	CmpLog.SysTypes.RTS_IEC_HANDLE	Дескриптор открытого логгера

3.8. Функция LogAdd

Функция **LogAdd** сохранена для совместимости со старыми версиями библиотеки. В настоящее время вместо нее следует использовать функцию **LogAdd2**.

3.9. Функция LogAdd2

Функция **LogAdd2** добавляет в логгер новую запись.

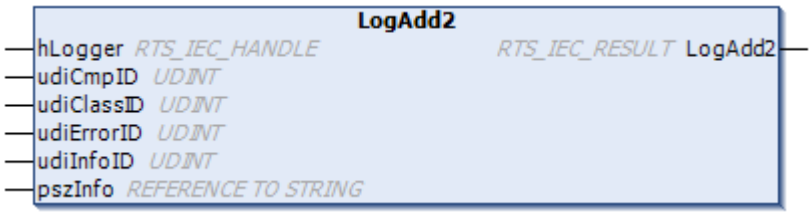


Рис. 3.6. Внешний вид функции **LogAdd2** на языке CFC

Табл. 3.6. Описание входов и выходов функции **LogAdd2**

Название	Тип	Описание
Входы		
hLogger	CmpLog.SysTypes.RTS_IEC_HANDLE	Дескриптор логгера. Может быть получен при вызове функции LogCreate или LogOpen. Также можно использовать псевдоним стандартного логгера
udiCmpID	UDINT	ID компонента, от имени которого будет опубликовано сообщение. См. подробнее в п. 3.12
udiClassID	UDINT	Класс сообщения (см. LogClass)
udiErrorID	UDINT	Код ошибки. Обычно используются коды из библиотеки **CmpErrors**
udiInfoID	UDINT	ID текста из файла описания устройства, используется для реализации мультиязычных сообщений (см. п. 3.13)
pszInfo	REFERENCE TO STRING	Текст сообщения. Поддерживаются только ASCII-символы. См. информацию о мультиязычных сообщениях в п. 3.13
Выходы		
LogAdd2	CmpLog.SysTypes.RTS_IEC_RESULT	Переменная, в которую будет записан код ошибки (см. библиотеку **CmpErrors**)

3.10. Функция LogClose

Функция **LogClose** закрывает логгер и обнуляет его дескриптор.

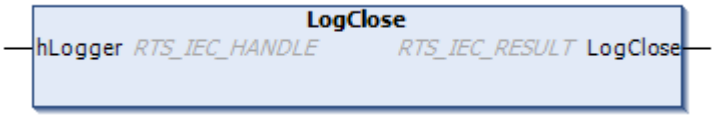


Рис. 3.7. Внешний вид функции **LogClose** на языке CFC

Табл. 3.7. Описание входов и выходов функции **LogClose**

Название	Тип	Описание
Входы		
hLogger	CmpLog.SysTypes.RTS_IEC_HANDLE	Дескриптор логгера. Может быть получен при вызове функции LogCreate или LogOpen
Выходы		
LogClose	CmpLog.SysTypes.RTS_IEC_RESULT	Переменная, в которую будет записан код ошибки (см. библиотеку **CmpErrors**)

3.11. Функция LogDelete

Функция **LogDelete** удаляет логгер.

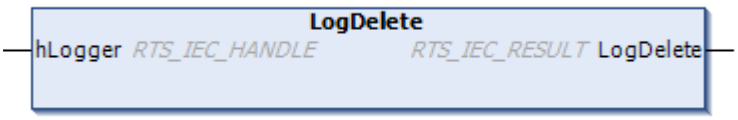


Рис. 3.8. Внешний вид функции **LogDelete** на языке CFC

Табл. 3.8. Описание входов и выходов функции **LogDelete**

Название	Тип	Описание
Входы		
hLogger	CmpLog.SysTypes.RTS_IEC_HANDLE	Дескриптор логгера. Может быть получен при вызове функции LogCreate или LogOpen
Выходы		
LogDelete	CmpLog.SysTypes.RTS_IEC_RESULT	Переменная, в которую будет записан код ошибки (см. библиотеку **CmpErrors**)

3.12. Регистрация компонента для получения ID

Одним из аргументов функции LogAdd2 является идентификатор компонента (**udiCmpID**), от имени которого публикуется сообщение. Предварительно компонент должен быть зарегистрирован в системе исполнения. Если же компонент не зарегистрирован или вместо идентификатора используется произвольное значение – то в логе вместо имени компонента будет отображаться это число.

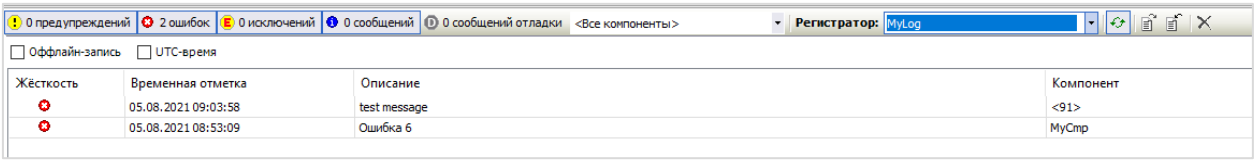


Рис. 3.9. Отличия сообщений от зарегистрированного и незарегистрированного компонента (см. столбец **Компонент**)

Для регистрации компонента используется библиотека **Component Manager**. Она содержит две функции для регистрации компонентов – **CMAddComponent** и **CMAddComponent2**. Отличие функций заключается в том, что для первой из них пользователь должен сам сформировать ID компонента, а при использовании второй – ID рассчитывается автоматически и возвращается по указателю.

Если ID рассчитывается пользователем, то он должен быть сформирован по следующей формуле:

```
// VENDOR_ID – идентификатор производителя (см. на вкладке Device – Информация,
// старшее слово параметра ID). Если производитель неизвестен – то используется
// значение 16#FFFF
// CMPID_IecCode – флаг, означающий, что компонент регистрируются из кода приложения
// LIBRARY_ID – идентификатор библиотеки (выбирается пользователем)

udiCmpId := VENDOR_ID OR Component_Manager.ComponentID.CMPID_IecCode OR LIBRARY_ID;
```

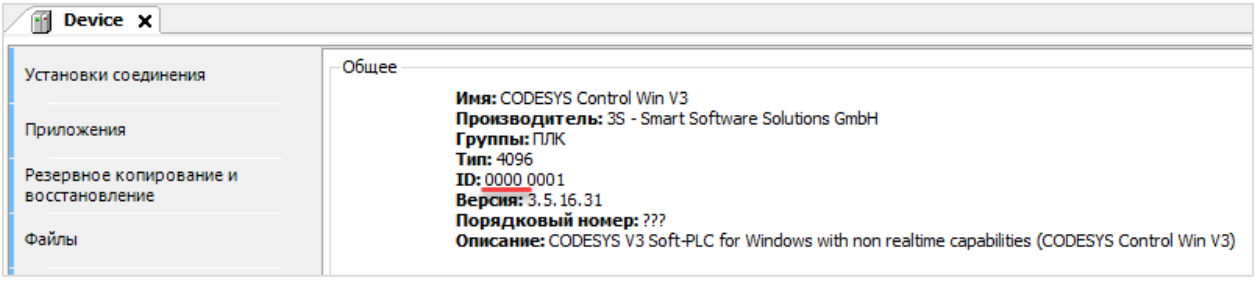


Рис. 3.10. Отображение **VENDOR_ID** на вкладке **Device - Информация**

Рис. 3.11. Внешний вид функции **CMAddComponent2** на языке CFC

Табл. 3.9. Описание входов и выходов функции **CMAddComponent2**

Название	Тип	Описание
Входы		
pszComponent	REFERENCE TO STRING	Имя, под которым регистрируется компонент (оно будет отображаться в логе)
udiVersion	UDINT	Версия компонента (например, 16#03050505 означает ‘3.5.5.5’)
udiCmpId	POINTER TO UDINT	Указатель, по которому будет записан ID компонента
pResult	SysTypes.RTS_IEC_RESULT	Переменная, в которую будет записан код ошибки (см. библиотеку **CmpErrors**)
Выходы		
CMAddComponent2	SysTypes.RTS_IEC_HANDLE	Дескриптор зарегистрированного компонента

Пример использования функции приведен в п. 3.14.

Отменить регистрацию компонента можно с помощью функции **CMRemoveComponent**.

3.13. Создание мультиязычных сообщений для логгера

Функция LogAdd2 позволяет добавить в лог только сообщение с кодировкой ASCII. Если требуется добавлять в лог сообщения с другими кодировками (например, Win1251) – то эти сообщения должны быть заранее добавлены в файл описания устройства или таргет-файл ПЛК.

```
<Strings namespace="RTSLog_MyCmp">
  <Language lang="ru">
    <String identifier="ID_00000006">Ошибка 6</String>
    <String identifier="ID_00000007">Ошибка 7</String>
  </Language>
  <Language lang="en">
    <String identifier="ID_00000006">Error 6</String>
    <String identifier="ID_00000007">Error 7</String>
  </Language>
</Strings>
```

Рис. 3.12. Пример объявления сообщений для логгера в файле описания устройства

- **Namespace** – имя компонента (состоит из префикса **RTSLOG_** и имени, под которым компонент регистрируются с помощью библиотеки Component Manager);
- **Lang** – язык сообщения (в среде программирования сообщения будут отображаться на языке, который выбран в ее настройках);
- **Identifier** – идентификатор сообщения в HEX с префиксом **ID_** (см. вход **udiInfoID** функции LogAdd2).

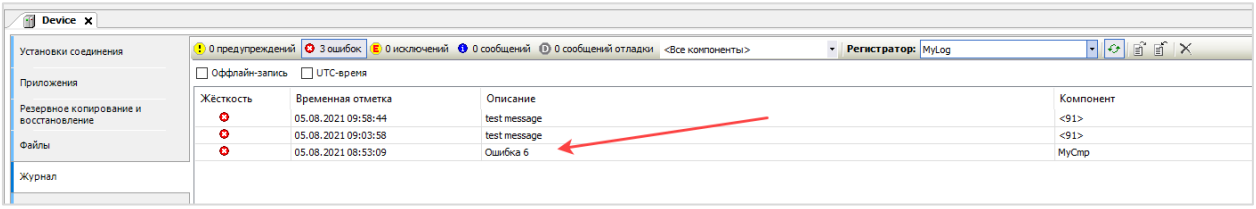


Рис. 3.13. Пример отображения в логе сообщений на кириллице

Также тексты сообщений могут быть выделены в отдельный файл, а в файле описания использоваться ссылка этот файл:

```
<Files namespace="local"> <!-- Note: The "Files" section is not necessary if all internationalized strings are in one external file.-->
    <Language lang="de">
        <File fileref="local" identifier="GenericLogStrings">
            <LocalFile>GenericLogStrings_de.xml</LocalFile>
        </File>
    </Language>
    <Language lang="en">
        <File fileref="local" identifier="GenericLogStrings">
            <LocalFile>GenericLogStrings_en.xml</LocalFile>
        </File>
    </Language>
</Files>
<Device>
    <DeviceInfo>
        ...
        <AdditionalFiles>
            <File key="ExternalStrings" name="local:GenericLogStrings">GenericLogStrings_en.xml</File>
        </AdditionalFiles>
    </DeviceInfo>
    ...
</Device>
```

Рис. 3.14. Пример использования внешнего файла с текстами сообщений

Пример такого файла: скачать (в файле используются доп. поля, которые могут быть нужны только при использовании Си-версии библиотеки)

3.14. Пример использования библиотеки CmpLog

Ниже приведен пример использования библиотеки **CmpLog** для создания логгера и записи в него сообщений. В примере используются библиотеки **CmpLog** и **Component Manager**. Ссылка на архив проекта для виртуального контроллера **CODESYS Control Win V3**, созданный в **CODESYS V3.5 SP16 Patch 3**: скачать

В примере используется модифицированный таргет-файл с версией **3.5.16.31**, который включает в себя тексты сообщений для событий с ID 6 и 7 (см. рис. 3.12).

```
PROGRAM PLC_PRG
VAR
          // команда создания логгера
          xCreateLogger:     BOOL;
          // команда записи в лог
          xWriteToLog:       BOOL;

          // дескриптор компонента
          hCmp:              CmpLog.SysTypes.RTS_IEC_HANDLE;
          // ID компонента
          udiCmpId:          UDINT;

          // дескриптор логгера
          hLog:              CmpLog.SysTypes.RTS_IEC_HANDLE;
          // настройки логгера
          stLogOptions:      CmpLog.LogOptions;
          // результат выполнения операции или код ошибки
          dwResult:          CmpLog.SysTypes.RTS_IEC_RESULT;
END_VAR


// команда создания логгера
IF xCreateLogger THEN
          // настраиваем логгер
          stLogOptions.bEnable         := 1;
          stLogOptions.szName          := 'MyLog';
          stLogOptions.uiType          := CmpLog.LogTypes.LT_STD;
          stLogOptions.uiFilter        := CmpLog.LogClass.LOG_ALL;
          stLogOptions.iMaxEntries     := 1000;

          // создаем логгер
          hLog := CmpLog.LogCreate(ADR(stLogOptions), ADR(dwResult) );
          // закрываем логгер
          dwResult := CmpLog.LogClose(hLog);

          // регистрируем компонент с именем MyCmp
          hCmp := Component_Manager.CMAddComponent2('MyCmp', 16#0305101E, ADR(udiCmpId),
ADR(dwResult) );
          xCreateLogger := FALSE;

END_IF

// команда записи в лог
IF xWriteToLog THEN

          // открываем логгер
          hLog     := CmpLog.LogOpen('MyLog', ADR(dwResult) );
          // добавляем в лог сообщение
          // оно также будет отображано во всплывающем окне (так как LOG_USER_NOTIFY)
          // если в файле описания устройства прописан текст для ID_00000006 - то в лог будет
выведен этот текст (на языке, выбранном в настройках среды программирования)
          // если в файле описания отсутствуют тексты - то будет выведен текст 'default text for
error 6'
          dwResult := CmpLog.LogAdd2(hLog, udiCmpId, CmpLog.LogClass.LOG_ERROR OR
CmpLog.LogClass.LOG_USER_NOTIFY, CmpErrors.Errors.ERR_FAILED, 6, 'default text for error 6');
          // закрываем логгер
          dwResult := CmpLog.LogClose(hLog);
          xWriteToLog := FALSE;

END_IF
```

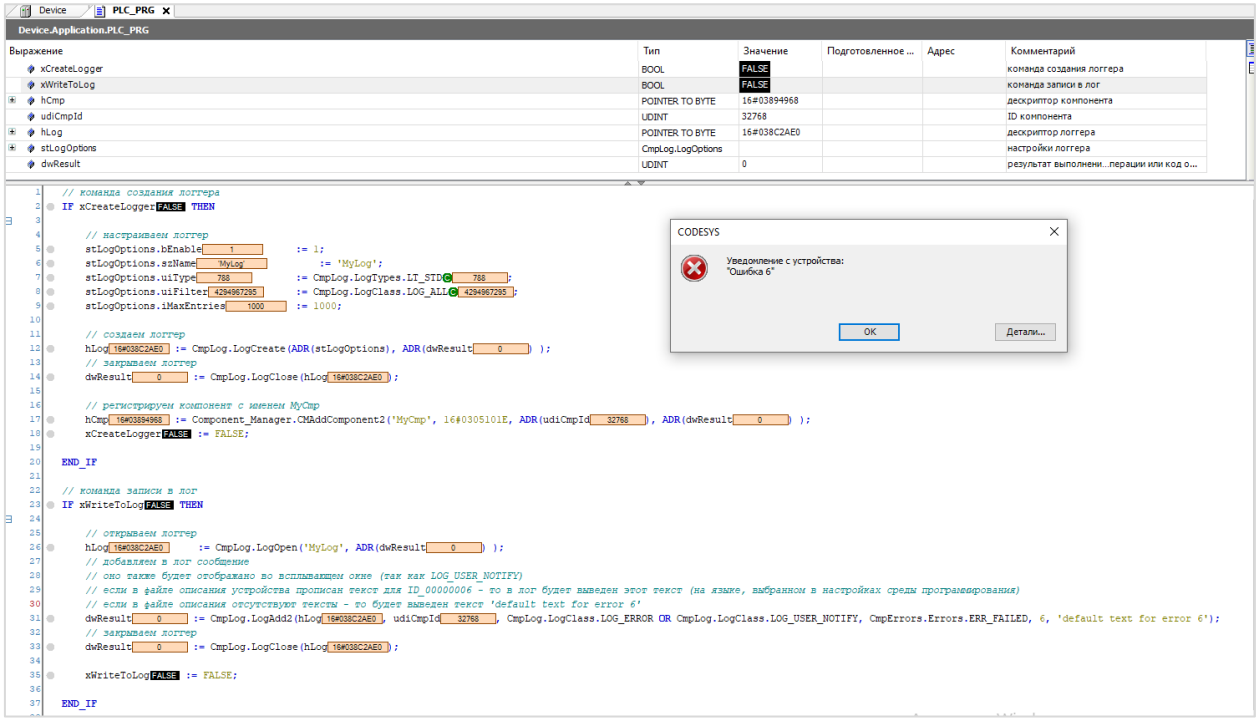


Рис. 3.15. Работа с примером – всплывающее окно с сообщением

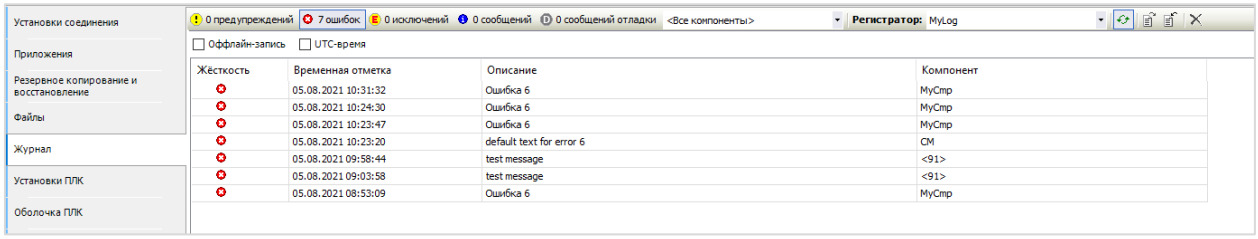


Рис. 3.16. Работа с примером – отображение сообщений на вкладке **Device – Журнал**

4. Библиотека CmpLogAsync

Библиотека **CmpLogAsync** используется для асинхронной работы с логгером. В библиотеку входит только один ФБ с двумя публичными методами и одна структура. Описание на библиотеку и пример использования отсутствуют – и нет уверенности, что она вообще полностью реализована и работоспособна.

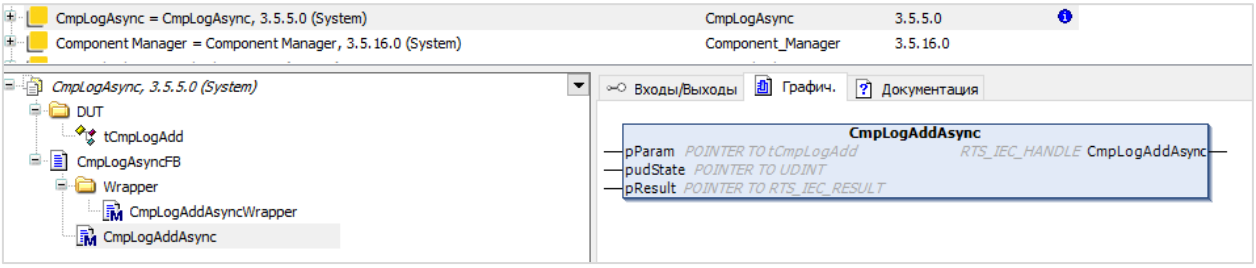


Рис. 4.1. Содержимое библиотеки **CmpLogAsync**

5. Публикация сообщений в лог от имени компонента визуализации

Можно опубликовать сообщение в стандартный логгер от имени компонента визуализации с помощью системной функции **VisuElems.Visu_Output** (соответственно, в проекте должна присутствовать визуализация).

Пример:

```
VAR
        xWriteToLog:            BOOL;
        sMessage:               STRING := 'Hello, world!';
END_VAR

IF xWriteToLog THEN
        VisuElems.Visu_Output(sMessage, VisuElems.LogClass.LOG_INFO);
        xWriteToLog:=FALSE;
END_IF
```

Первый аргумент функции – это текст сообщения, второй – его класс.

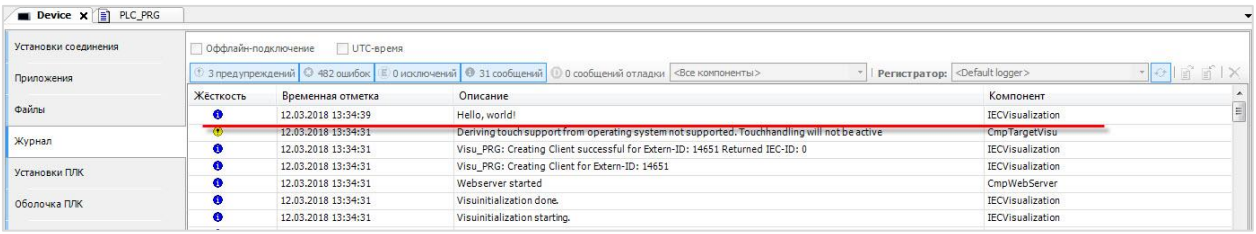


Рис. 5.1. Вывод сообщения в лог с помощью функции **VisuElems.Visu_Output**

6. Отправка сообщений на syslog-сервер

В случае необходимости можно настроить отправку сообщений логгера на syslog-сервер.

Для этого необходимо в конфиг-файле CODESYS:

1. Для нужного логгера добавить бэкенд с `ClassId=0x135`

2. Добавить секцию

```
[SysSocket]
; укажите IP-адрес и порт вашего syslog-сервера
sysLog.IPAddress=10.2.8.133
sysLog.Port=514
```